> このページはコミュニティーの尽力で英語から翻訳されました。MDN Web Docs コミュニティーについてもっと知り、仲間になるにはこちらから。

@media

`@media` は CSS のアットルールで、1 つまたは複数のメディアクエリーの結果に基づいて、スタイルシートの一部を適用するために使用することができます。これによってメディアクエリーを指定し、そのメディアクエリーがコンテンツの使用される端末に一致する場合にのみ、文書に CSS のブロックを適用することができます。

> **メモ:** JavaScript では、 `@media` を使用して作成されたルールは、 CSS オブジェクトモデルの `CSSMediaRule` インターフェイスによってアクセスすることができます。

構文

`@media` アットルールは、コードの最上位に配置したり、他の条件付きグループアットルールの中に入れ子にして配置したりすることができます。

```
/* コードの最上位 */
@media screen and (min-width: 900px) {
  article {
    padding: 1rem 3rem;
  }
}

/* 他の条件付きアットルールの中にネスト */
@supports (display: flex) {
  @media screen and (min-width: 900px) {
    article {
      display: flex;
    }
  }
}
```

メディアクエリーの構文について詳しくは、メディアクエリーの使用を参照してください。

解説

メディア種別

*メディア種別*は機器の全般的な分類を記述します。 `not` または `only` 論理演算子を使用する場合を除き、メディア種別は省略可能であり、暗黙で `all` と見なされます。

`all`

すべての機器に適合します。

`print`

ページのある資料や、画面に印刷プレビューモードで表示されている文書向けのものです。（これらの形式に特有の整形上の問題については、ページ付きメディアを参照してください。）

`screen`

主に画面向けのものです。

`speech`

音声合成向けのものです。

> **メモ:** CSS2.1 および Media Queries 3 では、その他のメディア種別 (`tty` , `tv` , `projection` , `handheld` , `braille` , `embossed` , `aural`) が定義されていましたが、これらは Media Queries 4 で非推奨となったため、使用すべきではありません。 `aural` タイプは `speech` に置き換えられましたが、これも同様です。

メディア特性

*メディア特性*は、ユーザーエージェント、出力装置、環境の特定の特徴を記述します。 メディア特性式は、その存在や値を調べるもので、完全に任意です。それぞれのメディア特性式は、括弧で囲む必要があります。

`any-hover`

入力メカニズムで、ユーザーが要素上でのホバーを使用することができるものがあるか。 Media Queries Level 4 で追加。

`any-pointer`

入力メカニズムにポインティングデバイスがあるか、もしそうであればどれだけ正確なものであるか。 Media Queries Level 4 で追加。

`aspect-ratio`

ビューポートの幅対高さのアスペクト比。

`color`

出力機器の色コンポーネントあたりの色のビット数、または端末がカラーでなければゼロ。

`color-gamut`

ユーザーエージェントおよび出力機器が対応しているおよその色の範囲。 Media Queries Level 4 で追加。

`color-index`

出力機器の色参照表の項目数、または端末がそのような表を使用していないのであればゼロ。

`device-aspect-ratio`

出力機器の幅対高さのアスペクト比。 Media Queries Level 4 で非推奨。

`device-height`

出力機器のレンダリング面の高さ。 Media Queries Level 4 で非推奨。

`device-width`

出力機器のレンダリング面の幅。 Media Queries Level 4 で非推奨。

`display-mode`

ウェブアプリのマニフェストの `display` メンバーで指定されているアプリケーションの表示モード。 Web App Manifest 仕様書 で定義。

`forced-colors`

ユーザーエージェントがカラーパレットを制限しているかどうかを検出。 Media Queries Level 5 で追加。

`grid`

出力機器はグリッドとビットマップ画面のどちらを使用するか。

height

ビューポートの高さ。

hover

主要な入力メカニズムで、ユーザーが要素上でのホバーを使用することができるか。 Media Queries Level 4 で追加。

inverted-colors

ユーザーエージェントまたはその下の OS が色を反転しているか。 Media Queries Level 5 で追加。

monochrome

出力機器のモノクロフレームバッファーにおけるピクセルあたりのビット数、または端末がモノクロでなければゼロ。

orientation

ビューポートの向き。

overflow-block

ビューポートをブロック軸方向にあふれたコンテンツを出力機器がどのように扱うか。 Media Queries Level 4 で追加。

overflow-inline

ビューポートをインライン軸方向にあふれたコンテンツがスクロールできるか。 Media Queries Level 4 で追加。

pointer

主要な入力メカニズムがポインティングデバイスであるか、もしそうであればどれだけ正確なものであるか。 Media Queries Level 4 で追加。

prefers-color-scheme

ユーザーが明るいまたは暗い色遣いを望んでいるかどうかを検出。 Media Queries Level 5 で追加。

prefers-contrast

ユーザーがシステムに隣り合う色のコントラスト量を増加または減少させるよう要求したかどうかを検出。 Media Queries Level 5 で追加。

`prefers-reduced-motion`

ユーザーがページであまり動きを望まないかどうか。 Media Queries Level 5 で追加。

`resolution`

出力機器のピクセル密度。

`scripting`

出力機器のスキャン処理方式。 Media Queries Level 5. で追加。

`update`

どれだけの頻度で出力機器がコンテンツの表示を変更できるか。 Media Queries Level 4 で追加。

`width`

スクロールバーの幅を含むビューポートの幅。

論理演算子

論理演算子 `not`, `and`, `only` を使うと、複雑なメディアクエリーを構成することができます。 また、複数のメディアクエリーをカンマで区切って 1 つのルールにまとめることもできます。

`and`

複数のメディア特性を 1 つのメディアクエリーに結合する際に使用されます。クエリーが `true` になるためには、結合させた各機能が `true` を返すことが必要です。 また、メディア特性とメディア種別を結合する際にも使用されます。

`not`

メディアクエリーを反転するために使用され、クエリーが `false` を返す場合に `true` を返します。 カンマで区切られたクエリーのリストの中にある場合は、適用された特定のクエリーのみを反転します。 `not` 演算子を使用する場合は、メディア種別も指定しなければなりません。

> **メモ:** レベル 3 では、個々のメディア特性式を否定するために `not` キーワードを使用することはできず、メディアクエリー全体のみを否定することしかできません。

`only`

クエリー全体が一致する場合にのみスタイルを適用します。 これは、古いブラウザーが選択したスタイルを適用できないようにするのに便利です。 `only` を使用しない場合、古いブラウザーは `screen and (max-width: 500px)` というクエリーを `screen` と解釈し、クエリーの残りの部分を無視して、すべての画面にそのスタイルを適用してしまいます。 `only` 演算子を使用する場合は、メディア種別も指定しなければなりません。

`,` (カンマ)

カンマは、複数のメディアクエリーを 1 つのルールにまとめるために使用されます。 カンマで区切られたリストの各クエリーは、他のクエリーとは別に扱われます。 したがって、リスト内のクエリーのいずれかが `true` であれば、メディア文全体が `true` を返すことになります。 言い換えれば、リストは論理的な `or` 演算子のように動作します.

アクセシビリティの考慮

サイトの文字の大きさを調整している人に合わせられるようには、メディアクエリーで `<length>` が必要な時には `em` を使用してください。

`em` および `px` はどちらも有効な単位ですが、 `em` はユーザーがブラウザーのテキストの寸法を変更した場合によりうまく動作します。

また、ユーザーの使い勝手を向上させるために Level 4 のメディアクエリーを使用することを検討してください。例えば、 `prefers-reduced-motion` はユーザーがシステムにアニメーションや動きの量を最小化するよう要求していることを検出します。

セキュリティ

メディアクエリーはユーザーが作業する端末の能力—およびその先にある、特性および設計—を調べる関係上、端末を識別する「指紋」を構築するために不正利用されたり、ユーザーにとって望ましくない観点のものに分類されるものに利用されたりする可能性があります。

この可能性のため、ブラウザーは返値がコンピューターを正確に識別するために使用することを防ぐために、何らかの方法で返値を偽造することがあります。ブラウザーは、この分野で他の手段を提供する場合もあります。例えば、 Firefox で「フィンガープリントの採取」をブロックしている場合、多くのメディアクエリーは実際の端末の状態を表す値ではなく既定値を報告します。

形式文法

```
@media =
  @media <media-query-list> { <stylesheet> }
```

例

print および screen メディア種別の検査

```
@media print {
  body { font-size: 10pt; }
}

@media screen {
  body { font-size: 13px; }
}

@media screen, print {
  body { line-height: 1.2; }
}

@media only screen
  and (min-width: 320px)
  and (max-width: 480px)
  and (resolution: 150dpi) {
    body { line-height: 1.4; }
}
```

Media Queries Level 4 では、範囲を受け付ける特性を検査する際に、以下のようなより簡潔にメディアクエリーを記述できる新しい範囲の構文が導入されました。

```
@media (height > 600px) {
    body { line-height: 1.4; }
}

@media (400px <= width <= 700px) {
    body { line-height: 1.4; }
}
```

他の例については、メディアクエリーの使用をご覧ください。

仕様書

Specification
Media Queries Level 4 # media-descriptor-table
CSS Conditional Rules Module Level 3 # at-media

ブラウザーの互換性

関連情報

- メディアクエリーの使用
- JavaScript では `@media` は CSS オブジェクトモデルの `CSSMediaRule` インターフェイスを通じてアクセスすることができます。
- Mozilla 拡張メディア特性
- WebKit 拡張メディア特性

This page was last modified on 2022年11月30日 by MDN contributors.